BROOK+ SAMPLE

Cholesky Factorization

1 Introduction

This Brook+ sample computes the Cholesky factorization of a positive definite matrix. A positive definite matrix, **A**, can be represented as a product of a lower triangular matrix and its transpose:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^{T}$$

For a 2x2 matrix,

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

Simplifying the RHS we get,

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11}^{2} & \\ l_{21} * l_{11} & \end{bmatrix} \begin{bmatrix} l_{11} * l_{21} & \\ l_{21} * l_{11} + & l_{22}^{2} \end{bmatrix}$$

The entries of L can be computed as follows:

1. $l_{11} = \text{sqrt}(a_{11})$
2. $l_{21} = a_{21} / l_{11}$
3. $l_{22} = \text{sqrt}(a_{22} - l_{21}^{2})$

The last step can be separated into:

3.a) $a_{22} = (a_{22} - l_{21}^{2})$

3.b) $l_{22}, = \text{sqrt}(a_{22})$

These steps for a 2 x 2 matrix can be generalized for any positive-definite matrix of dimensions ***m* x *n*** as an iterative algorithm consisting of three basic steps:

1. square-root,
2. normalization, and
3. sub-matrix update.

These are applied iteratively to progressively smaller regions of the original matrix, **A**. In the naïve version, the algorithm processes individual elements of the matrix **A**. This algorithm, according to Jung et al. [1], is given below and shown in Figure 1.

```
for k = 1 to n-1 do
    A(k, k) = sqrt(A(k, k))                                  // Square root Step
    A(k+1:n, k) = A(k+1:n, k)/A(k, k)                        // Normalization Step
    for j = k+1 to n do
        A(j, k+1) = A(j, k+1) - A(j, 1:k)T x A(k+1, 1:k)     // Sub-matrix update step
    end for
end for
A(n, n) = sqrt(A(n, n))
```

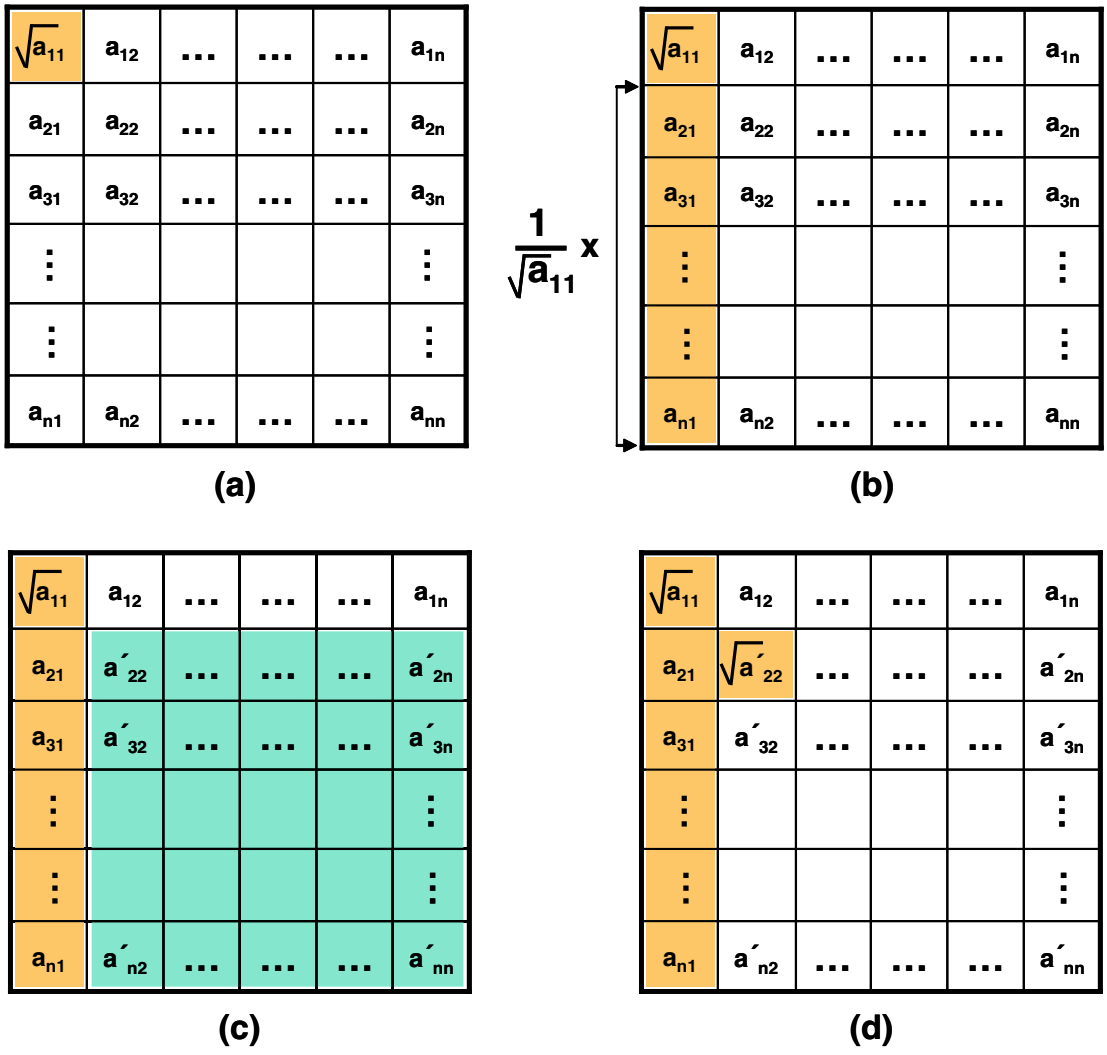

Figure 1 Elementwise Computation of the Cholesky Factorization

For Figure 1:

a. Square root, $l_{11} = \text{sqrt}(a_{11})$.
b. Normalize a_{21} (highlighted in black).
c. Update sub-matrix a_{22} (highlighted in black).
d. Repeat steps on sub-matrix starting with square-root.

2 Blocked Cholesky Factorization

The elementwise algorithm is inefficient because it processes only one column in every iteration. We can achieve a significant performance gain by considering entire matrix blocks at a time. In block-partitioned form, the Cholesky factorization of a positive matrix **A** of size *m* x *m* is:

$$\begin{bmatrix} \mathbf{A_{11}} & \mathbf{A^T_{12}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} \end{bmatrix} = \begin{bmatrix} \mathbf{L_{11}} & \mathbf{0} \\ \mathbf{L_{21}} & \mathbf{L_{22}} \end{bmatrix} \begin{bmatrix} \mathbf{L^T_{11}} & \mathbf{L^T_{21}} \\ \mathbf{0} & \mathbf{L^T_{22}} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{L_{11}L^T_{11}} & \mathbf{L_{11}L^T_{21}} \\ \mathbf{L_{21}L^T_{11}} & \mathbf{L_{21}L^T_{21} + L_{22}L^T_{22}} \end{bmatrix}$$

This shows that $\mathbf{L_{11}}$ is the Cholesky factor of $\mathbf{A_{11}}$. Assume that a routine Cholesky computes the Cholesky factorization of a matrix using the elementwise computation described earlier. We then can have the following iterative algorithm for block-partitioned Cholesky factorization of a matrix:

1. Square Root Step: $\mathbf{L_{11}}$ = Cholesky($\mathbf{A_{11}}$)
2. Normalization Step: $\mathbf{L_{21}} = \mathbf{A_{21}}\ (\mathbf{L^T_{11}})^{-1}$
3. Sub-matrix Update Step: $\mathbf{A_{22'} = A_{22} - L_{21}\ L_{21}^T = L_{22}\ L_{22}^T}$
4. Repeat steps 1 to 3 on residual matrix $\mathbf{A_{22'}}$.

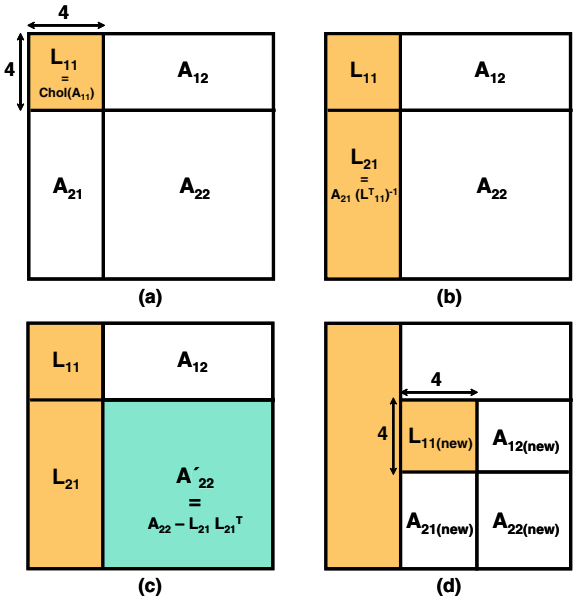

Figure 2 Blocked Cholesky Factorization

For Figure 2:

a. Block partitions of matrix A
b. Square Root Step: L_{11} = Cholesky(A_{11})
c. Normalization Step: $L_{21} = A_{21}\ (L^T_{11})^{-1}$
d. Sub-matrix Update Step $A_{22'} = A_{22} - L_{21}\ L_{21}^T = L_{22}\ L_{22}^T$

For the second pass, the input to the algorithm is the updated sub-matrix $\mathbf{A_{22'}}$ (yellow region).

3 Brook+ SIMD Implementation

For the Brook+ sample in the SDK, we choose a block size of 4x4. This can be stored conveniently in four float4 variables. We provide a kernel for each step of the algorithm. The square root step is achieved by computing an elementwise Cholesky factorization of the 4x4 matrix. The normalization kernel performs two tasks:

1. It computes the inverse-transpose of the 4x4 Cholesky factor $\mathbf{L_{11}}$.
2. It uses the result to perform the normalization by matrix multiplication.

The sub-matrix update kernel consists of a matrix-multiplication step followed by matrix subtraction.

4 Data Transfer

Our blocked Cholesky factorization sample follows a multi-pass approach that iterates over progressively smaller regions of the input matrix with each pass. Specifically, at the end of each pass, we are only interested in the residual updated sub-matrix **A22'** (see Figure 2). We use a ping-pong strategy to facilitate data transfer from one pass to the next. Two streams, ***A*** and ***L***, are declared; the input matrix is initially copied to both. Brook+ enforces the rule that the same stream cannot be bound simultaneously for reading and writing. We always bind ***L*** for writing; thus, the partial results always remains in the ***L*** stream. At the end of each pass, we copy the sub-domain in ***L*** corresponding to the updated sub-matrix **A22'** back to ***A***.

5 Performance

The performance of Brook+ Cholesky Factorization was tested on a system with the AMD Athlon™ 64 X2 Dual Core Processor CPU and an ATI Radeon™ HD 4800 Series GPU. Figure 3 shows a comparison of the time taken to compute the Cholesky Factorization of matrices of varying sizes for the GPU versus that of the CPU.

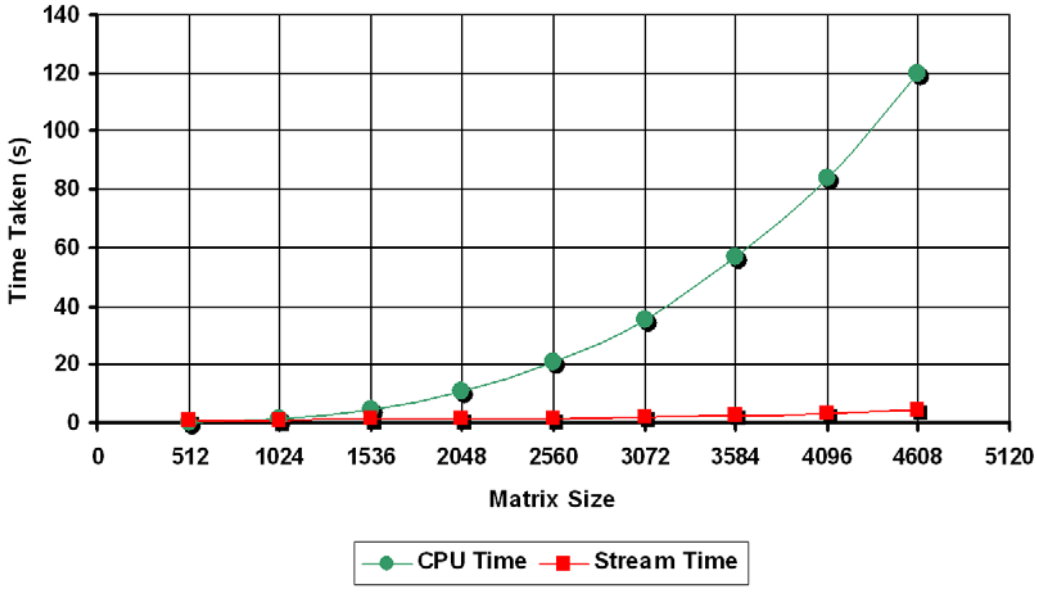

Figure 3 Performance of Brook+ Cholesky Factorization: Stream Processor vs CPU

Figure 4 shows the speedup achieved over the reference CPU implementation. The Radeon™ 4800 Series can achieve speedups of over 25 for this sample implementation.

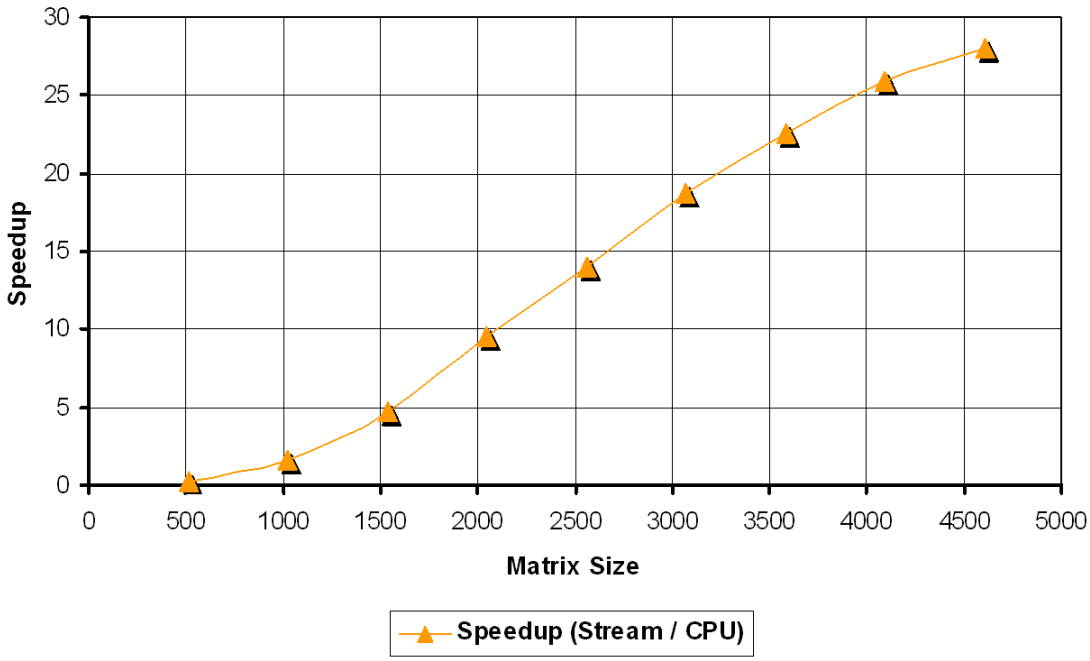

Figure 4 Speedup of Cholesky Factorization: Stream Processor vs CPU Implementation

6 References

1. Jin Hyuk Jung, “Cholesky Decomposition and Linear Programming on a GPU,” Scholarly Paper Directed by Dianne P. O'Leary (2006).

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For Stream Computing:
URL: http://ati.amd.com/technology/streamcomputing
Questions: streamcomputing@amd.com
Developing: streamdeveloper@amd.com
Forum: http://forums.amd.com/devforum/categories.cfm?catid=38